Interactive Lighting and Material Design System for Cyber Worlds

Kei Iwasaki
Faculty of Systems Engineering
Wakayama University
Wakayama, Japan
iwasaki@sys.wakayama-u.ac.jp

Yoshinori Dobashi
Graduate School of
Information Science and Technology
Hokkaido University
Sapporo, Japan
doba@ime.ist.hokudai.ac.jp

Tomoyuki Nishita
Graduate School of Frontier Sciences
The University of Tokyo
Tokyo, Japan
nis@is.s.u-tokyo.ac.jp

***Abstract*—Interactive rendering under complex real world illumination is essential for many applications such as material design, lighting design, and virtual realities. For such applications, interactive manipulations of viewpoints, lighting, BRDFs, and positions of objects are beneficial to designers and users. This paper proposes a system that acquires complex, all-frequency lighting environments and renders dynamic scenes under captured illumination, for lighting and material design applications in cyber worlds. To capture real world lighting environments easily, our method uses a camera equipped with a cellular phone. To handle dynamic scenes of rigid objects and dynamic BRDFs, our method decomposes the visibility function at each vertex of each object into the occlusion due to the object itself and occlusions due to other objects, which are represented by a nonlinear piecewise constant approximation, called cuts. Our method proposes a compact cut representation and efficient algorithm for cut operations. By using our system, interactive manipulation of positions of objects and realtime rendering with dynamic viewpoints, lighting, and BRDFs can be achieved.**

***Keywords*-image-based lighting, interactive rendering, material design**

I. Introduction

Capturing complex real world lighting environments and rendering objects under the captured lighting environments are beneficial to many applications in cyber worlds such as lighting design, interior design, and web shopping. For example, rendered images of goods (e.g. furniture and clothes) under the illumination where the goods are actually used are beneficial to customers in web shoppings.

In recent years, image-based lighting techniques, which use the captured image as an environment map for the lighting, have been widely studied. Although image-based lighting techniques can render realistic images, real-time rendering using image-based lighting is quite difficult due to the expensive computational cost of the integration over the hemisphere of lighting directions at each calculation point. Sloan et al. proposed a Precomputed Radiance Transfer (PRT) method that precomputes the transfer function at each vertex and achieved real-time rendering of static scenes under environment lighting [1]. Although several methods that extend the PRT method have been proposed ([2], [3], [4]), interactive rendering of dynamic scenes of rigid objects with dynamic BRDFs remains a challenging problem. Several applications, such as interior design and lighting design, require the interactive manipulations of viewpoints, lighting, the positions of objects, and BRDFs. Moreover, it is quite difficult to capture lighting environments and create environment maps since several special devices such as omnidirectional cameras and spherical mirror balls are required.

This paper proposes a system that captures the lighting environments and renders dynamic scenes under captured lighting environments. Our system estimates the lighting environments by taking two photographs with a camera equipped with a cellular phone. Then our system renders rigid objects using the estimated illumination allowing users to change the viewpoint, lighting, positions of rigid objects, and BRDFs. Previous PRT methods have several limitations, such as the fixed viewpoint and lighting [9], static scenes [6] and the editable BRDFs are limited and the precomputed data size is quite large [10]. In contrast, the proposed method can edit arbitrary BRDFs and the data size required for the proposed method is compact since no precomputed BRDF data is necessary. To deal with dynamic scenes, our method precomputes the shadowing effects of a rigid object at sample points around the object. In the rendering process, the visibility function at each vertex is calculated by combining the precomputed shadow effects of other objects and self-shadow due to the object itself. The shadowing effects recorded at sample points and the self-shadowing effect at each vertex are represented by a combination of piecewise constant functions called cuts [5].

The contributions of the proposed method are as follows:

- complex, all-frequency lighting environments can be acquired easily by using a cellular phone
- interactive rendering with dynamic viewpoints, lighting, positions of objects, and BRDFs
- a compact representation and efficient operation algorithm for cuts

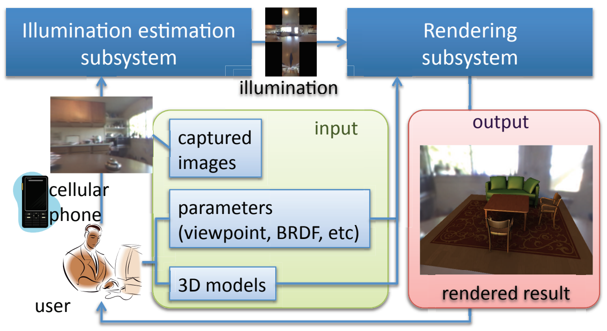


Figure 1. Overview of our system.

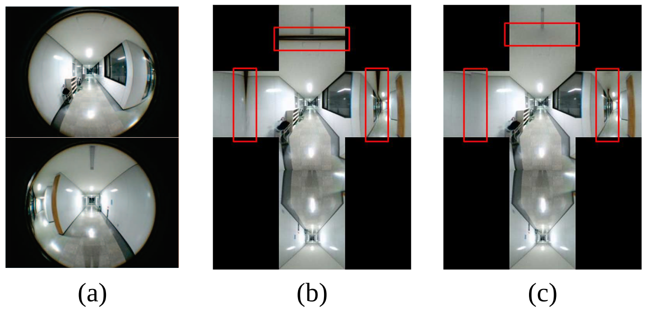


Figure 2. (a) images captured by using the fish-eye lens. (b) reflections of the black frame of the fish-eye lens are shown in red rectangles. (c) reflections of the black frame of the fish-eye lens are almost removed.

II. Previous Work

PRT methods calculate the transfer function at each vertex and represent the transfer function with a linear combination of spherical harmonics functions [1] and wavelets [2] in the preprocess. Although these methods can achieve real-time rendering using environment lighting, scenes and BRDFs are fixed to calculate the transfer functions in the preprocess. Zhou et al. [3] and Sun et al. [4] proposed the PRT-based methods to handle dynamic scenes, but are limited to fixed BRDFs.

In recent years, real-time rendering methods using shadow-maps for environment lighting have been proposed [7], [8] These methods, however, approximate the environment lighting with a small number of area light sources, and evaluate the BRDF with a small number of sampling directions, resulting in inaccurate calculations of radiances.

Several methods have been proposed to edit BRDFs. Lawrence et al. proposed an inverse shade tree method that decomposes BRDFs into editable 1D functions for BRDF editing [11]. This method, however, is limited to a point light source. Colbert et al. [12] proposed a BRDF editing method called BRDF shop, which can edit only Ward BRDFs. Ben-Artzi et al. proposed a real-time BRDF editing method for complex, all-frequency lighting for static viewpoint and lighting [9]. Sun et al. developed an interactive relighting method for dynamic BRDFs with global illumination [10]. This method, however, assumes that the scene is static and the precomputed BRDF data is quite large. Akerlund et al. precomputed visibility cuts that enable BRDF editing in the rendering process [5]. Cheslack-Postava et al. extended visibility cuts to handle per-pixel shading [6]. Recently, Wang et al. proposed a real-time rendering method for dynamic, spatially-varying BRDFs under all-frequency lighting environments [13]. Although these methods can change BRDFs in the rendering process, these methods assume that the scenes are static.

III. System Overview

The overview of our system is shown in Fig. 1. Our system consists of two subsystems: *illumination estimation subsystem* and *rendering subsystem*. Firstly, to capture the complex real world lighting, the user takes photographs of the environment that the user wants to create in the cyber world. Our system assumes that the device to take the photograph is a camera with a cellular phone since cellular phones are widely used and the performances of the cameras of the cellular phones are advancing, though the other devices such as the digital cameras also can be used. Secondly, the illumination estimation subsystem estimates the lighting environment by using the captured photographs. The illumination estimation subsystem calculates the source lighting of the environment and stores it in a cubemap. Then the rendering subsystem renders a scene that the user wants to design. The rendering subsystem allows the user to change the viewpoint, lighting, BRDFs of the objects, the positions of the objects interactively.

IV. Illumination Estimation Subsystem

An input to the illumination estimation subsystem is an information of the lighting environments of the scene that the user wants to create in the cyber world. To capture the whole view of the scene, our method uses a camera with a fish-eye lens, which is easily obtained. By using the fish-eye lens, the user takes only two photographs of the scene (front side and back side of the scene) as shown in Fig. 2(a).

Next, the illumination estimation subsystem creates High Dynamic Range (HDR) images from the two photographs, which are stored in Low Dynamic Rage (LDR) images, to capture complex, all-frequency lighting environments. To create HDR images from LDR images, different exposure images are required [16]. To do this, our method utilizes the auto exposure bracketing option of the camera of the cellular phone, which can automatically take three or more shots with a different exposure.

By using the HDR images, the source lighting incident onto the scene is estimated. However, as shown in Fig. 2(b),

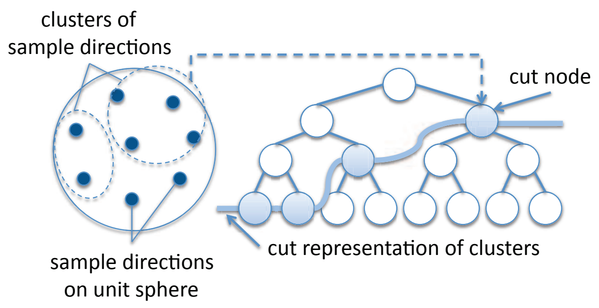

clusters of sample directions
cut node
sample directions on unit sphere
cut representation of clusters

Figure 3. Cut representation of clustered sample directions.

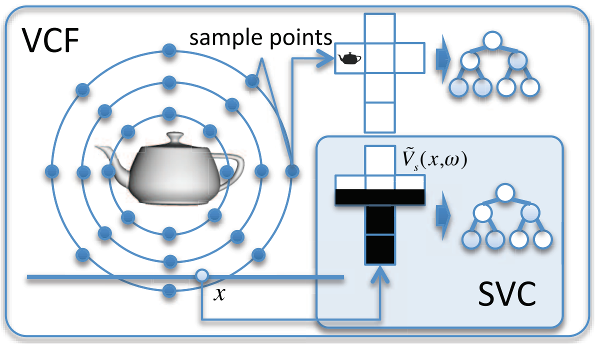


Figure 4. Self Visibility Cut (SVC) and Visibility Cut Field (VCF).

the frame of the fish-eye lens is captured in the images, which may lead to artifacts in the rendered images.

To address this problem. our method eliminates the reflections of the frame of the fish-eye lens by using the seamless cloning method in [15]. A rectangle area that does not include the reflections of the frame is selected and is composed into the images seamlessly as shown in Fig. 2(c). The edited images are stored in a cubemap and used in the rendering subsystem.

V. Rendering Subsystem

The rendering subsystem outputs the rendered images of the scene using the estimated illumination. Similar to the previous methods [3], [4], our method assumes the dynamic scenes of rigid objects and direct illumination. Fast global illumination calculation of dynamic scenes with dynamic BRDFs is a challenging problem and we leave it to future work. The outgoing radiance $B(x, \omega_o)$ at each vertex x in viewing direction ω_o is calculated from:

$$B(x, \omega_o) = \int_\Omega L(\omega)V(x, \omega)f_r(\omega, \omega_o)\cos\theta d\omega, \qquad (1)$$

where Ω is the directions on unit sphere, $L(\omega)$ is the source lighting represented by an environment map, $V(x, \omega)$ is the visibility function, $f_r(\omega, \omega_o)$ is the BRDF, $\cos\theta$ is calculated by the inner product of the normal at vertex x and the incident direction ω and clamped to 0. The proposed method samples the incident directions on the unit sphere into tens of thousands directions to capture complex, all-frequency lighting. Eq. (1) is calculated from:

$$B(x, \omega_o) = \sum_{j=1}^{S} L(\omega_j)V(x, \omega_j)f_r(\omega_j, \omega_o)\cos\theta\Delta_j, \qquad (2)$$

where S is the number of sample directions, Δ_j is the solid angle corresponding to the sample direction ω_j. Since it is impractical to calculate Eq. (2) for many sample directions, our method calculates Eq. (2) by clustering sample directions as:

$$B(x, \omega_o) \approx \sum_{k=1}^{N} l_k v_k f_r(\omega_k, \omega_o), \qquad (3)$$

where N is the number of clusters, l_k is calculated from $l_k = \int_{\Omega_k} L(\omega)d\omega$, Ω_k is the solid angle corresponds to cluster C_k, ω_k is the representative direction of cluster C_k, v_k is calculated from $v_k = \frac{1}{|\Omega_k|}\int_{\Omega_k} V(x, \omega)\cos\theta d\omega$.

The clustering method is based on the binary tree structure [5]. As shown in Fig. 3, each leaf node corresponds to each sample direction and stores the value of visibility function. The inner node that represents the cluster stores the average of values stored at two children nodes and the error value due to the clustering. The values of the inner nodes are calculated in a bottom-up manner. Then our method selects the nodes whose error is smaller than the threshold by traversing the binary tree from the root node. The visibility function V is represented by a set of clusters called *cuts*. The visibility function V is calculated from the product of the self-visibility V_s due to the object itself and the occlusion V_i due to other object i as:

$$V(x, \omega) = V_s(x, \omega)\prod_{i=1} V_i(x, \omega). \qquad (4)$$

For rigid objects, the self-visibility can be precomputed at each vertex, and the occlusion due to each object can be recorded at sample points around the object in the preprocess. In our method, the product of the self-visibility V_s and the cosine term is represented with $\tilde{V}_s(x, \omega)$. $\tilde{V}_s(x, \omega)$ is precomputed and clustered at each vertex. The cut of $\tilde{V}_s(x, \omega)$ is referred to as Self Visibility Cut (SVC). The occlusions recorded at sample points around each object are also precomputed and represented with cuts. These cuts are referred to as Visibility Cut Field (VCF) (see Fig. 4).

The outgoing radiance at each vertex is calculated by using Eq. (2). The cluster value v_k is calculated by using SVC and VCF. BRDF $f_r(\omega_k, \omega_o)$ is directly evaluated on the fly. The outgoing radiance at each vertex is calculated by summing the product $l_k v_k f_r(\omega_k, \omega_o)$ over the cut nodes.

A. Implementation details

1) Precomputation: The proposed method precomputes SVC at each vertex of each object, and VCF of each object.

In the binary tree structure for clustering, each leaf node corresponds to each sample direction on the unit sphere. The sample directions are calculated by distributing points uniformly on the unit sphere using a point repulsion algorithm [14]. In the proposed method, the number of sample directions is set to 32,768 ($= 2^{15}$), which gives good results. First, SVC at each vertex of each object is calculated. At each vertex, rays of sample directions are traced and the values of $\tilde{V}_s(x, \omega_j)$ are stored at leaf nodes. The clustering and selection of cut nodes are similar to [5].

Next, VCF for each object is calculated. The sample points that record the occlusion due to the object are generated on the concentric spheres whose center is the center of the object (see Fig. 4). The occlusions at each sample point are calculated by tracing rays of sample directions. The occlusion $V_i(x_s, \omega)$ due to object i at sample point x_s is a binary function. That is, if a ray from x_s in direction ω intersects object i, $V_i(x_s, \omega)$ returns 0, otherwise $V_i(x_s, \omega)$ returns 1. The occlusion $V_i(x_s, \omega)$ is sampled at each sample direction ω_j and stored at each leaf node. To render sharp shadows, our method clusters the two child nodes only if the values of two child nodes are the same. The clustered nodes are stored as the cut nodes. VCF stores the cuts at all sample points.

2) Cut representation: Since VCF stores all the cuts at densely sampled points around each object, the size of the VCF is quite large. To address this problem, the proposed method reduces the data size of VCF by using a new representation of cuts. For the VCF, each cut node stores the binary value (0 or 1) of the cut node and the position of the cut node. In the previous methods [5], [6], the position of the cut node is represented by using an index (see Fig. 5). The index of the node requires 2bytes for 2^{15} directions. Since the binary value of the cut node can be represented by only 1bit, the data size of the index of the node is dominant in the data size of the cut nodes.

The proposed method exploits the property that every path from the root node to a leaf node contains exactly one cut node. By utilizing this property and ordering the cut nodes from the leftmost node in the tree, the position of the cut node can be identified by using the height of the node (the number of the edges from the leaf node). The proposed method represents the position of the node by using its height instead of the index (see Fig. 5). The proposed method can represent the node position using only 4bits since the height of the node ranges from 0 to 15.

3) Operations of height-based cuts: In the rendering process, the clustered value v_k of the visibility function at each vertex is calculated by using SVC and VCF. We explain the operations of multiple cuts such as the product and the interpolation. The operation of cut nodes is performed only when the cut nodes satisfy the relationship of descendent and ascendant. Therefore, the operand cut nodes are traversed to keep the relationship. The proposed method prepares

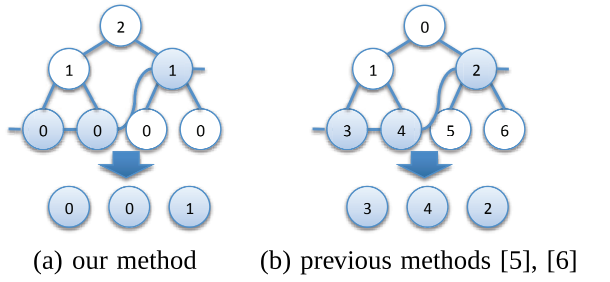


Figure 5. Cut representation. (a) cut nodes are represented by its height. The number in the node represents its height. (b) cut nodes are represented by index. The number in the node represents its index.

counters to determine whether the operand cut nodes satisfy the relationship or not. Let n_j be the operand node of j-th cut pointed by pointer p_j, h_j be the height of node n_j, c_j be the counter of cut j, and h_{min} be the minimum height in all operand nodes. The operation processes of m cuts ($1 \leq j \leq m$) are as follows:

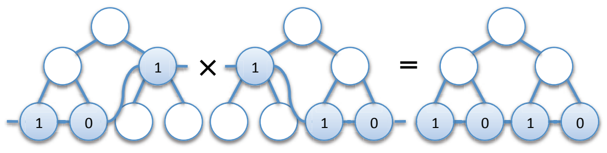
Step 1 Initialize the counter c_j of j-th cut as zero and set pointer p_j to point to the first cut node.

Step 2 Perform operations, such as product and interpolation of nodes n_j. Store the value at the node whose height is the minimum height h_{min} in $(h_1, h_2, \cdots, h_m)$.

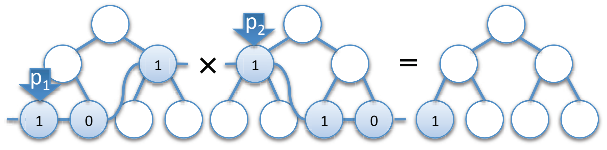
Step 3 (a): If height h_j is equal to h_{min}, advance the pointer p_j and set counter c_j to zero.
(b): Otherwise (i.e. $h_j > h_{min}$), add $2^{h_{min}}$ to the counter c_j. If $c_j \geq 2^{h_j}$, advance pointer to the next node and set c_j to zero.

Step 4 Repeat from Step 2 to Step 3 until all pointers point to the last cut node.

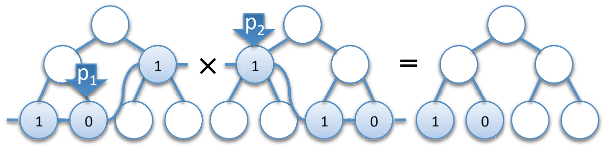
Fig. 6 shows the calculation process of the product of the cut nodes. In Fig. 6, the number in the node represents the value of the cluster and the pointers are represented by arrows. As shown in Fig. 6(b), pointers p_1 and p_2 are set to point to each leftmost node, and c_1 and c_2 are initialized to zero (Step 1). The operation of pointed nodes is performed (Step 2). The result is stored in the node whose height is the minimum height h_{min} of the operand cut nodes. In this case the minimum height h_{min} is h_1. After the operation, pointer p_1 is advanced (Step 3(a)). For node n_2, since h_2 is greater than h_{min}, $2^{h_{min}}$ is added to counter c_2 (Step 3(b)). By comparing counter c_2 with 2^{h_2}, nodes n_1 and n_2 are tested whether to satisfy the relationship of descendant and ascendant. In Fig. 6(c), nodes are considered to satisfy the relationship since $c_2 = 1 < 2^{h_2}$. Then the operation of nodes is performed again (Step 2), and pointer p_1 is advanced (Step 3(a)). For node n_2, $2^{h_{min}}$ is added to counter c_2 (Step 3(b)). At this time, counter c_2 is equal to 2^{h_2} and nodes n_1 and n_2 do not satisfy the relationship of descendant and ascendant as shown in Fig. 6(d). Then pointer p_2 is advanced and c_2 is set to zero (Fig. 6(e)). By using our method, each cut node



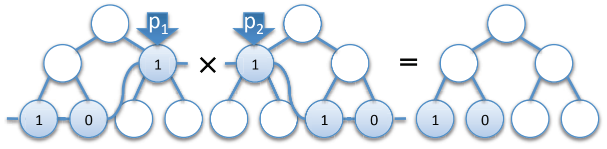
(a) result of product of two cuts



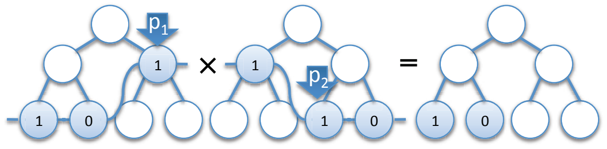
(b) counters are initialized as : $c_1 = 0$, $c_2 = 0$ (Step1). The product of two cut nodes is stored in the node whose height is $h_{min} = 0$ (Step 2),



(c) pointer p_1 is advanced since $h_1 = h_{min}$ (Step 3(a)). counter c_2 is added to 1.



(d) $c_2 = 2$ and c_2 is equal to 2^{h_2} (Step 3(b)).



(e) pointer p_2 is advanced and c_2 is set to 0.

Figure 6. Product of two cuts.

always satisfies the relationship of descendent and ascendant.

4) Rendering: In the rendering process, the outgoing radiance at each vertex is calculated from Eq. (3). The cluster value v_k of the visibility function is obtained by using SVC and VCFs. For the VCF of each object, eight sample points neighbor to the vertex are calculated. The cluster values of the occlusion function due to object i are interpolated by using the cuts stored at eight sample points. The cluster values of the visibility function $V(x, \omega_j)$ are calculated by multiplying the cluster values of the SVC with the interpolated cluster values of the VCFs. The cluster value of source lighting l_k for cluster C_k and the evaluated BRDF $f_r(\omega_k, \omega_o)$ are multiplied with v_k, and the product $l_k v_k f_r(\omega_k, \omega_o)$ is summed over all cut nodes.

B. Acceleration

Since the computational cost of radiance calculations using cuts is quite high, our method accelerates the radiance calculation by using visibility culling. Firstly, our method calculates only the radiances of visible vertices from the viewpoint. To detect the visible vertices, our method assigns ID to each triangle and renders the triangles by setting the ID to its color. Then by reading the pixel values from the framebuffer, IDs of the visible triangles are obtained. The vertices of the visible triangles are considered as the visible vertices. Secondly, for each vertex, our method culls the objects that do not contribute to the radiance of each vertex. Our method culls the objects whose bounding spheres are under the local horizon plane of the vertex. Finally, since the radiance calculation of each vertex can be parallelized, our method calculates the radiances of vertices in parallel by using multithread programming.

VI. Results

Fig. 7 shows the rendering results of the furniture scene with dynamic viewpoints, lighting, BRDFs and translating objects. Fig. 7(a) shows the furniture scene whose materials are all diffuse BRDFs. In Fig. 7(b), BRDFs of a chair, table, and sofa are changed to Phong BRDFs. Fig. 7(c) shows the rendering result of translating a chair and Fig. 7(d) is rendered with different viewpoint and lighting. Figs. 7(e) and (f) are rendered by using environment maps captured by using a cellular phone CASIO EXLIM W63CA (the resolution of the camera is 809Mpixels).

The rendering frame rate of translating objects is 7fps, and that for dynamic viewpoint, lighting and BRDFs is 42fps on a PC with Intel Core i7 Extreme 965 and an NVIDIA GeForce GTX 295. The number of vertices in Fig. 7 is 27,000 and the precomputed data size of SVC and VCFs is 270MB. Please note that the data size is smaller than [3], [10]. To generate sample points for VCF, 32 concentric spheres of the bounding sphere are distributed and 6×32^2 sample points are distributed on each concentric sphere. That is, 196,608 sample points are generated for each VCF. The data size of SVC and VCF using the previous method [6] is 687MB. The average cut sizes of SVCs and VCFs of this example are 428 and 710, respectively. That is, the proposed method can reduce the precomputed data about 61%.

Fig. 8 shows a comparison of rendered images with different number of sample points of VCF. Fig. 8(a) is the reference image of the static scene of a teapot and a floor. Figs. 8(b) to (e) show the images with different number of sample points of VCF of the teapot. The concentric spheres for VCF are uniformly distributed between $0.2r$ to $10r$, where r is the radius of the bounding box of the object. Fig. 8(b) (64 concentric spheres and 6×64^2 sample points on each concentric sphere) is indistinguishable to the reference image (Fig. 8(a)). However, the required data size is quite large for this very simple scene. In Figs. 8(d) (16 concentric spheres and 6×16^2 sample points) and (e) (8 concentric spheres and 6×8^2 sample points), shadows due to the teapot are blurred due to low sampling of VCF. In our examples, we obtain plausible results with 32 concentric spheres with 6×32^2 sample points of VCF as shown in Fig. 8(c).

VII. Conclusion and Future Work

We have developed a prototype system that captures lighting environments and renders scenes under captured illumination interactively. Our illumination estimation subsystem can calculate all-frequency, complex lighting environments by only taking photographs using a camera of a cellular phone. Our rendering subsystem can render dynamic scenes with dynamic viewpoint, lighting and BRDFs interactively. The visibility functions are efficiently calculated by using SVC and VCF. We have proposed a new cut representation based on the node height, which can represent the cut data compactly, and an efficient operation method of cuts. Our system enables the users to change the positions of objects at interactive rates and to change the viewpoint, lighting and BRDFs in real-time.

The prototype system is a stand-alone system. In future work, we would like to extend our method to a server-client model. Moreover, in the illumination estimation subsystem, the elimination of the reflections of the frame of the fish-eye lens is performed by the user. We would like to develop an automatic method that can eliminate the reflections to reduce the user's burden.

Acknowledgment

This research was partially supported by Grant-in-Aid for Young Scientists (B), Japan Science and Technology Agency, and Research Foundation for Electrotechnology of Chubu. We thank Yuuki Miyamae and Ryota Nakanishi for the development of the prototype system.

References

[1] P.P. Sloan, J. Kautz, and J. Snyder, ”Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environents”, ACM Transactions on Graphics, Vol. 21, No. 3, pp.527-536, 2002.

[2] R. Ng, R. Ramamoorthi, and P. Hanrahan, ”All-Frequency Shadows using Non-Linear Wavelet Lighting Approximation”, ACM Transactions on Graphics, Vol. 22, No. 3, pp. 376-381, 2003.

[3] K. Zhou, Y. Hu, S. Liu, B. Guo, and H.Y. Shum, ”Precomputed Shadow Fields”, ACM Transactions on Graphics, Vol. 24, No. 3, pp. 1196-1201, 2005.

[4] W. Sun and A. Mukherjee, ”Generalized Wavelet Product Integral for Rendering Dynamic Glossy Objects”, ACM Transactions on Graphics, Vol. 25, No. 3, pp. 477-487, 2006.

[5] O. Akerlund, M. Unger, and R. Wang, ”Precomputed Visibility Cuts for Interactive Relighting with Dynamic BRDFs”, Proc. Pacific Graphics, pp. 161-170, 2007.

[6] E.C. Postava, R. Wang, O. Akerlund, and F. Pellacini, ”Fast, Realistic Lighting and Material Design using Non-linear Cut Approximation”, ACM Transactions on Graphics, Vol. 27, No. 5, Article 128, 2008.

[7] T. Annen, Z. Dong, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz, ”Real-time, All-frequency Shadows in Dynamic Scenes”, ACM Transactions on Graphics, Vol. 27, No. 3, Article 34, 2008.

[8] T. Ritschel, T. Grosch, M.H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, ”Imperfect Shadow Maps for Efficient Computation of Indirect Illumination”, ACM Transactions on Graphics, Vol. 27, No. 5, Article 129, 2008.

[9] A. Ben-Artzi, R. Overbeck, and R. Ramamoorthi, ”Real-time BRDF Editing in Complex Lighting”, ACM Transaction on Graphics, Vol. 25, No. 3, pp. 945-954, 2006.

[10] X. Sun, K. Zhou, Y. Chen, S. Lin, J. Shi, and B. Guo, ”Interactive Relighting with Dynamic BRDFs”, ACM Transactions on Graphics, Vol. 26, No. 3, Article 27, 2007.

[11] J. Lawrence, A. Ben-Artzi, C. DeCoro, W. Matusik, H. Pfister, R. Ramamoorthi, and S. Rusinkiewicz, ”Inverse Shade Trees for Non-Parametric Material Representation and Editing”, ACM Transactions on Graphics, Vol. 25, No. 3, pp. 735-745, 2006.

[12] M. Colbert, S. Pattanaik, and J. Krivanek, ”BRDF-Shop: Creating Physically Correct Bidirectional Reflectance Distribution Functions”, IEEE Computer Graphics and Applications, Vol. 26, No. 1, pp. 30-36, 2006.

[13] J. Wang, P. Ren, M. Gong, J. Snyder, and B. Guo, ”All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance”, ACM Transactions on Graphics, Vol. 28, No. 5, Article 133, 2009.

[14] G. Turk, ”Generating textures on arbitrary surfaces using reaction diffusion”, Computer Graphics, Vol. 25, No. 4, pp. 289-298, 1991.

[15] P. Perez, M. Gangnet, and A. Blake, ”Poisson Image Editing”, ACM Transactions on Graphics, Vol. 22, No. 3, pp. 313-318, 2003.

[16] P.E. Debevec and J. Malik, ”Recovering High Dynamic Range Radiance Maps from Photographs”, Proc. SIGGRAPH 1997, pp. 369-378, 1997.

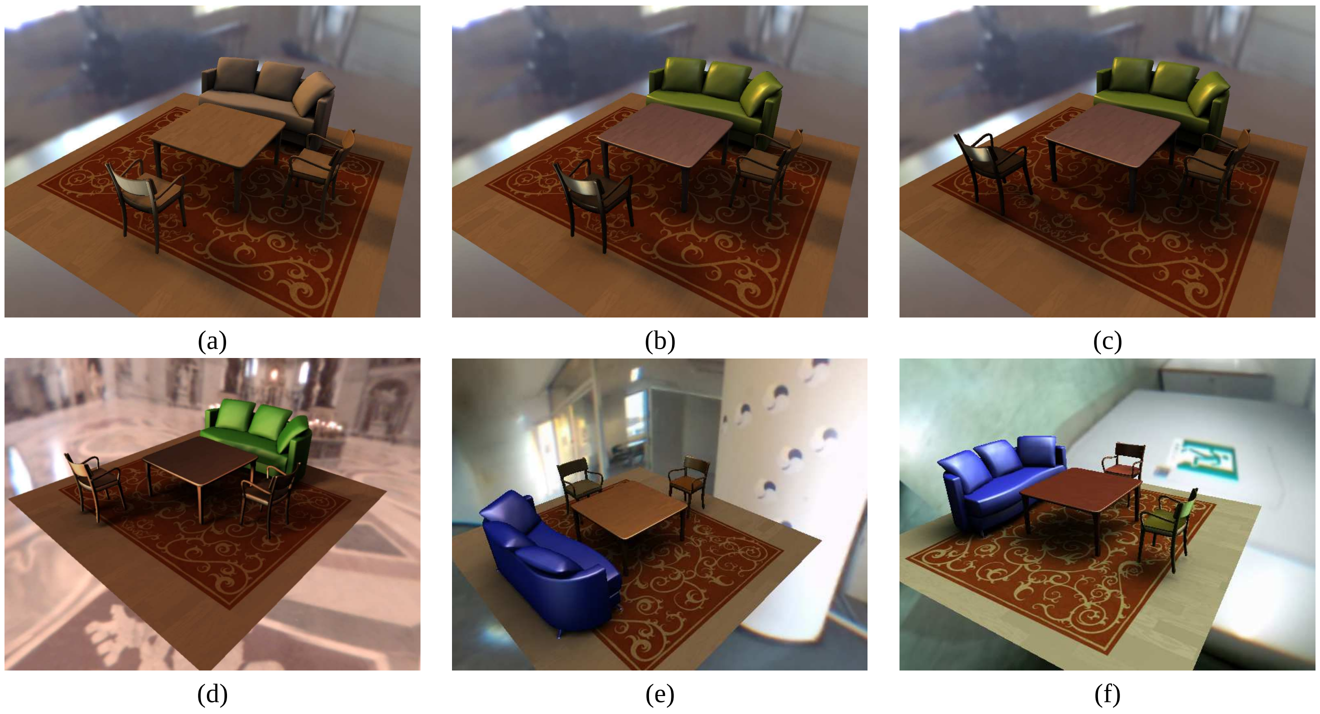


Figure 7. Rendering results using our system.

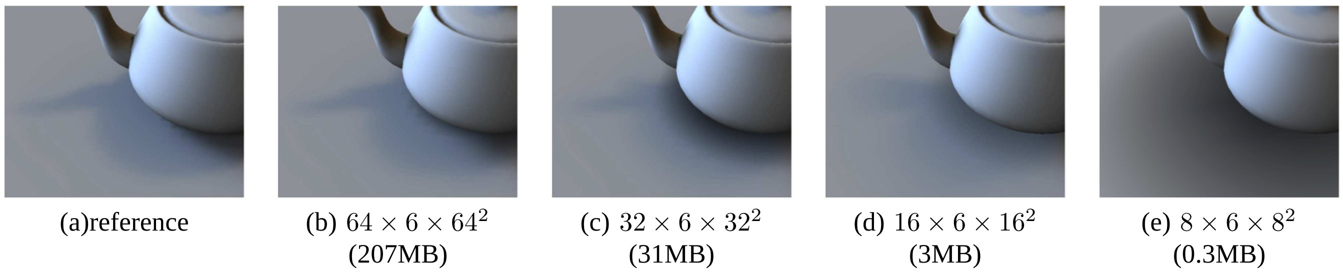

(a)reference
(b) $64 \times 6 \times 64^2$ (207MB)
(c) $32 \times 6 \times 32^2$ (31MB)
(d) $16 \times 6 \times 16^2$ (3MB)
(e) $8 \times 6 \times 8^2$ (0.3MB)

Figure 8. Comparison of the images with different number of sample points for VCF.